P2P e-Learning System and Its Squeak-based User Interface

Takao KAWAMURA, Ryosuke NAKATANI, and Kazunori SUGAHARA
Department of Information and Knowledge Engineering
Tottori University
4–101, Koyama-Minami
Tottori, JAPAN
{kawamura,rnakatan,sugahara}@ike.tottori-u.ac.jp

Abstract

In this paper, we present a novel framework for asynchronous Web-based training. The proposed system has two distinguishing features. Firstly, it is based on P2P architecture for scalability and robustness. Secondly, all contents in the system are not only data but also agents so that they can mark user's answers, can tell the correct answers, and can show some extra information without human instruction. We also present a prototype implementation of the proposed system on Maglog. Maglog is a Prolog-based framework for building mobile multi-agent systems we have developed. The user interface program of the proposed system is built on Squeak. Performance simulations demonstrate the effectiveness of the proposed system.

1. Introduction

The term e-Learning covers a wide set of applications and processes, such as Web-based training (hereafter we abbreviate as WBT), computer-based training, virtual classrooms, and digital collaboration. We are concerned with asynchronous WBT that allows the learner to complete the WBT on his own time and schedule, without live interaction with the instructor.

Although a large number of studies have been made on asynchronous WBT[3, 4], all of them are based on the client/server model. The client/server systems generally lack scalability and robustness. In the recent years, P2P research has grown exponentially. Although the current P2P systems are famous for its file sharing ability, and the consequent legal problems, P2P systems are gradually proving to be a very promising area of research. Because they have potential for offering a decentralized, self-sustained, scalable, fault tolerant and symmetric network of computers providing an effective balancing of storage and bandwidth resources.

In this paper, we present a novel framework for asynchronous WBT. The proposed system has two distinguishing features. Firstly, it is based on P2P architecture and every user's computer plays the role of a client and a server. Namely, while a user uses the proposed e-Learning system, his/her computer (hereafter we refer to such a computer as a node) is a part of the system. It receives some number of contents from another node when it joins the system and has responsibility to send appropriate contents to requesting nodes. Secondly, each content in the system is not only data but also an agent so that it can mark user's answers, tell the correct answers, and show some extra information without human instruction.

This paper is organized in 8 sections. We describe the supposed situation for the proposed system in Section 2. In Section 3, we describe our design goals. In Section 4, we describe the design of the proposed system and a prototype implementation of the system. In Section 5, we describe the user interface program built on Squeak. In Section 6, we present performance simulations. In Section 7, we briefly review related work. Finally, in Section 8, we describe some concluding remarks.

2. Supposed Situation

As mentioned in the previous section, we focus on asynchronous WBT, that is to say, a user can connect to the proposed e-Learning system anytime and anywhere he/she wants. Once connection is established, the user can obtain exercises one after another through specifying categories of the required exercises. User's answers

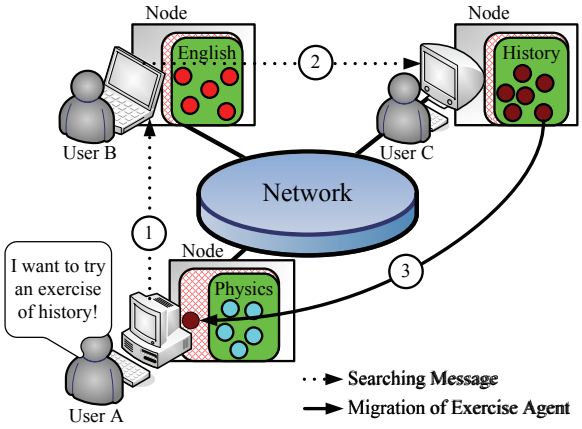

Figure 1. Proposed e-Learning system.

for each exercise are marked as correct or incorrect right away. Extra information may be provided for each answer, which can be viewed when the correct answer is shown.

While a user uses the proposed e-Learning system, his/her computer is a part of the system. Namely, it receives some number of categories and exercises in them from another node when it joins the system and has responsibility to send appropriate exercises to requesting nodes.

The important point to note is that the categories a node has are independent of the categories in which the node's user are interested as shown in Figure 1. Figure 1 illustrates that user A's request is forwarded at first to the neighbor node, next forwarded to the node which has the requested category.

3. Basic Concepts

As mentioned above, the proposed system has two distinguishing features. Firstly, it is based on P2P architecture. Secondly, each exercise is not only data but also an agent so that it can mark user's answers, tell the correct answers, and show some extra information about the exercise. In this section, we describe these features in detail.

3.1. P2P Aspect

All exercises in the proposed system are classified into categories such as "Mathematics / Expression / Equation", "English / Grammar", and "History / Rome", etc.

When the proposed system begins, one initial node has all categories in the system. When another node joins the system, it is received some number of categories from the initial node. The categories are distributed among all nodes in the system according as nodes join the system or leave the system.

We would like to emphasize that in existing P2P-based file sharing systems such as Napster[8], Gnutella[2], and Freenet[1] each shared file is owned by a particular node. Accordingly, files are originally distributed among all nodes. On the other hand, the categories in the proposed system are originally concentrated. Consequently, when a new node joins the system, not only location information of a category but the category itself must be handed to the new node. Considering that, the P2P network of the proposed system can be constructed as a CAN[11].

A CAN has a virtual coordinate space that is used to store $(key, value)$ pairs. To store a pair (K_1, V_1), key K_1 is deterministically mapped onto a point P in the coordinate space using a uniform hash function. The corresponding $(key, value)$ pair is then stored at the node that owns the zone within which the point P lies. In the proposed system, we let each category be a key and let a set of exercises belonging to the category be the corresponding value.

3.2. Mobile Agent Aspect

Generally, in addition to service to show an exercise, a WBT server provides services to mark the user's answers, tell the correct answers, and show some extra information about the exercise. Therefore, for the proposed system which can be considered a distributed WBT system, it is not enough that only exercises are distributed among all nodes. Functions to provide the above services also must be distributed among all nodes. We adopt mobile agent technology to achieve this goal. Namely, an exercise is not only data but also an agent so that it can mark user's answers, tell the correct answers, and show some extra information about the exercise.

In addition, mobile agent technology is applied to realize the migration of categories, that is, each category is also an agent in the proposed system.

4. Design and Implementation

We have implemented a prototype of the proposed system on Maglog that is a Prolog-based framework for building mobile multi-agent systems we have developed[6].

As shown in Figure 2, a node consists of the following agents and a user interface program. The components

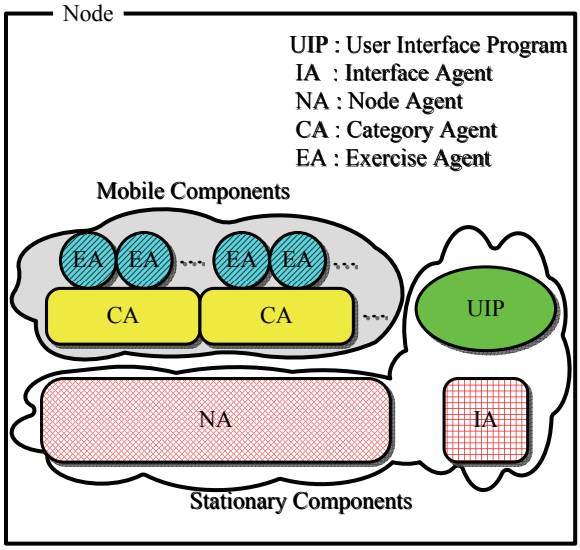

Figure 2. Architecture of a node.

of a node are divided into two type, those that move to other node referred as mobile components in Figure 2, and those that keep their station referred as stational components in Figure 2.

Node Agent There is one node agent on each node. It manages the zone information of a CAN and forwards messages to the category agents in the node.

Category Agent Each category agent stands for a unit of a particular subject. It manages exercise agents in itself and sends them to the requesting node.

Exercise Agent Each exercise agent has a question and functions to mark user's answers, tell the correct answers, and show some extra information about the exercise. These data are formatted in HTML.

Interface Agent There is one interface agent on each node. It is an interface between the user interface program and other agents.

Agents communicate with other agents through 'field's provided by Maglog framework. A field is kind of a preemptive queue. Roughly speaking, the above-mentioned four kinds of agents execute a message dispatch loop. Each message to an agent is queued into the field owned by the agent. The user interface program also communicates with the interface agent through a field via XML-RPC[14]. Table 1 shows a partial summary of message types.

Table 1. Partial summary of message types.

Dispatcher	Type	Description
Node	join	To join the system.
Node	leave	To leave the system.
Node	update	To update the neighbour's zone information.
Node	request	To get an exercise agent.
Category	add	To add an exercise agent.
Category	go	To go to another node.
Category	send	To send an exercise agent.
Category	receive	To receive an exercise agent.
Exercise	go	To go to the requesting node.
Exercise	show	To show a question.
Exercise	mark	To mark a user's answer.
Exercise	answer	To show the correct answer.
Exercise	info	To show the extra information.
Interface	retrieve	To get an exercise agent.
Interface	release	To let an exercise agent go home.
Interface	arrived	To be notified the arrival of an exercise agent.

5. User Interface Program

5.1. Features

As mentioned above, the user interface program of the proposed system has been developed through extending Scamper which is a simple web browser runs in Squeak[5].

Figures 3, 4, 5, 6, and 7 are screen-shots of the user interface program. The main window of it consists of three panes. Firstly, the button pane includes three buttons to get exercises. Secondly, the category pane shows categories of exercises. Thirdly, the exercise pane shows an exercise. Categories are classified into a hierarchical tree structure, as shown in Figure 4. By clicking the left button of a mouse on the category, a user can select it. After selection of the category, a user can obtain an exercise belonging to the category by clicking the left button of a mouse on one of buttons in the button pane. After a while an appropriate exercise agent comes from some node and the user can try the question as shown in Figure 5. The

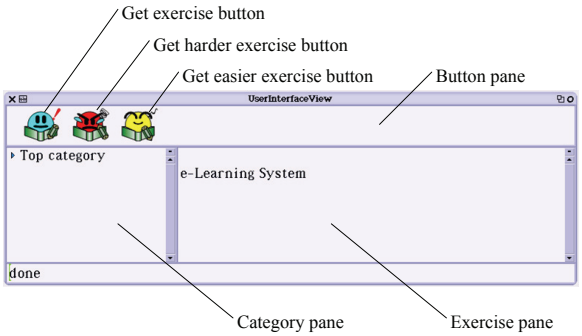

Figure 3. The window of the user interface program.

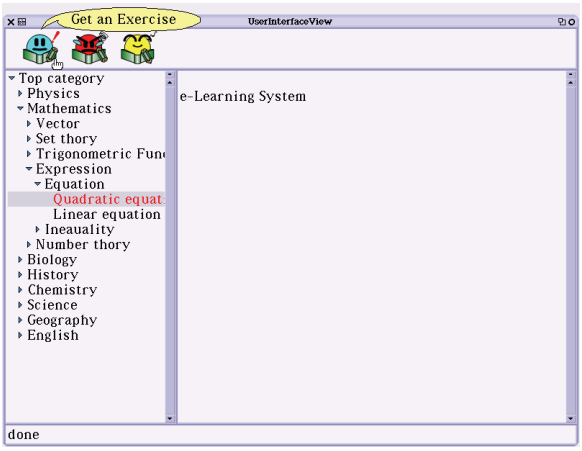

Figure 4. An exercise can be obtained by clicking the right button of a mouse on a buttons in the button pane.

user can require to mark his/her answer anytime by clicking the submit button. Figure 6 shows an example result of marking. Figure 7 shows the correct answers and extra information about the exercise that are shown by clicking the answer button.

Through the interativeness of Squeak, a user "do it" a sentence in the answer window which includes the correct answers and extra information about the exercise that he/she is trying shown as Figure 7.

In addition to, a user can operate the user interface program through not only GUI but also sending messages to the object that realize the user interface program. Figure 8 shows an example of sending messages to the user interface object in a workplace.

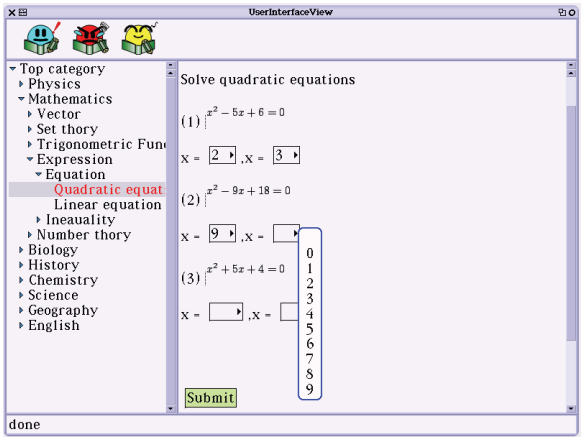

Figure 5. An appropriate exercise agent comes from some node and is tried by the requested user.

5.2. Implementation

The user interface program consists of the following classes.

UserInterface This class is "Model" part in MVC. An instance of this class receives messages from a user directly or from an instance of the UserInterface-View through the dependency mechanism. It responds to user's requests by delegating them to an instance of the Exercise class or the interface agent via an instance of the AgentInterface class.

UserInterfaceView This class corresponds to both "View" part and "Control" part in MVC. It sends user's requests to an instance of the UserInterface class and displays exercises or answers in a window.

Exercise An instance of this class, which is a proxy for an exercise agent, delegates messages to the agent.

AgentInterface This class provides a communication channel between objects in Squeak and agents in Maglog using XML-RPC.

Figure 9 shows the relationship among classes.

6. Performance Simulation

This section presents performance simulations obtained from a prototype implementation of the proposed system described in the previous section.

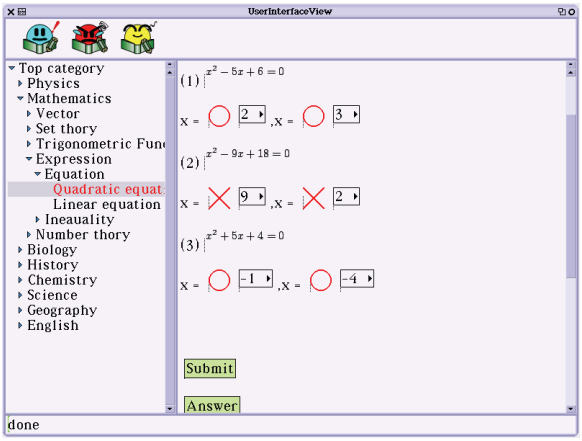

Figure 6. User's answers are marked as correct or incorrect by clicking the submit button.

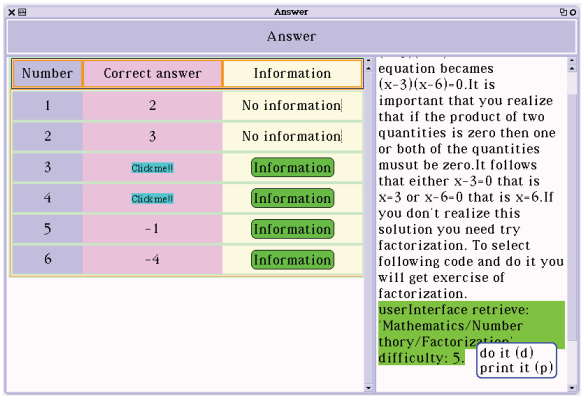

Figure 7. Correct answers and extra information are shown by clicking the answer button on the screen shown as Figure 6.

Table 2. Experimental Conditions.

Number of Nodes	8
Number of Categories	8
Number of Exercises/Category	50
Searching Frequency [times/sec]	$\frac{1}{12}, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}$

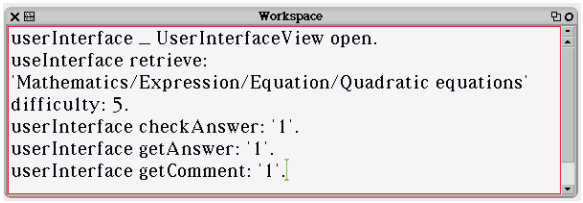

Figure 8. Workspace.

The experimental environment consists of 8 PCs with Intel Pentium4 2.4GHz processor and 512MB of RAM, and all the PCs are running on GNU/Linux (kernel version is 2.2.26) operating system. The physical network layout is shown in Figure 10.

We measured the searching latency in the experimental environment under the conditions shown in Table 2. All nodes send searching requests at the same time and exercises to be searched are selected randomly. We compared distributed and concentrated systems where these terms are defined as follows:

Distributed System Each node has one category.

Concentrated System One node has all categories and the rest nodes have no category.

It must be noted that the distributed system represents the proposed system in which all categories have ideal distribution. The concentrated system is equivalent to an ordinary WBT system.

Simulations are carried out with the time interval of 600 seconds. Each Simulation is repeated 10 times and the average of those is reported in Figure 11. Naturally, the higher searching frequency is, the larger searching latency is. Figure 11 shows that searching latency grows rapid in the concentrated system, while it grows slowly in the distributed system. In other words, the result suggests that the proposed e-Learning system has higher scalability than ordinary concentrated WBT systems have.

7. Related Works

A great deal of effort has been made on agent-based systems[15, 7, 13, 12]. However, these technologies provide support for agent collaboration and communication but lack support for P2P technology. Therefore, there are few agent-based P2P applications. PeerDB[10] is one of them, however agent technology is only used to assist query processing while in the proposed e-Learning system it is used not only for interactiveness but also for migration of the functionality of the system.

Edutella is P2P network for exchanging information about learning objects[9]. Edutella is based on RDF(Resource Description Framework), which is a framework for representing information in the Web. Con-

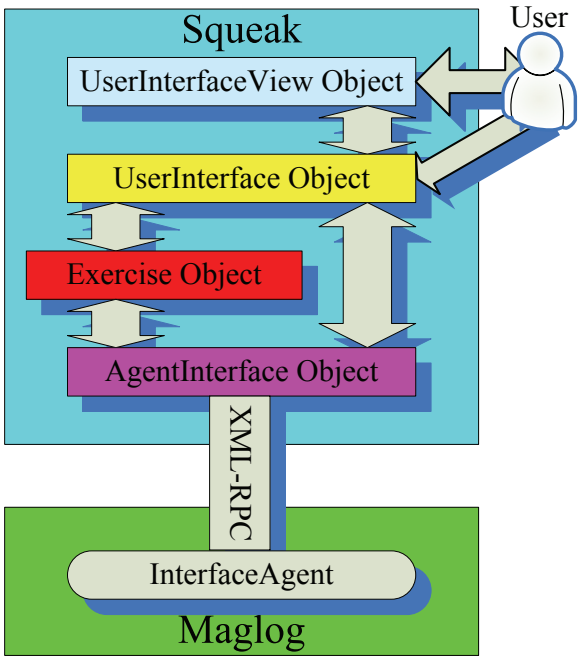

Figure 9. Relationship among classes.

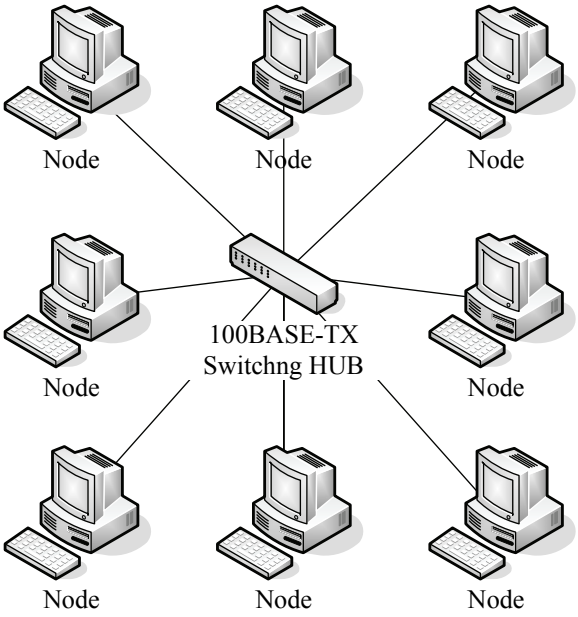

Figure 10. Experimental environment.

sequently, Edutella does not intend to receive user's response. In contrast, that is one of main goal of the proposed system and it is achieved through agent technology.

8. Conclusion

Since existing asynchronous WBT systems are based on the client/server model, they have problems of scalability and robustness. The proposed e-Learning system solves these problems in decentralized manner through both P2P technology and mobile agent technology. The user interface program of the proposed system is built on Squeak so that it obtains much interactiness and flexibility. Performance simulations suggest that the proposed e-Learning system has higher scalability than ordinary concentrated WBT systems have.

The expansion to provide popular functions of ordinary concentrated WBT systems for communication between the instructor and the learners and among the learners by means of email, BBS (Bulletin Board System) and 'Chat', in decentralized manner, is left for future work.

References

[1] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. http://freenetproject.org/freenet.pdf, 1999.

[2] Gnutella. http://welcome.to/gnutella/.

[3] D. Helic, H. Krottmaier, H. Maurer, and N. Scerbakov. Implementing project-based learning in wbt systems. In *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pages 2189–2196, November 2003.

[4] H. Homma and Y. Aoki. Creation of wbt server on digital signal processing. In *Proceedings of 4th International Conference on Information Technology Based Higher Education and Training*, July 2003. Marrakech, Morocco.

[5] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay. Back to the future: The story of squeak, a practical smalltalk written in itself. In *Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 318–326, 1997.

[6] T. Kawamura, S. Kinoshita, K. Sugahara, and T. Kuwatani. A logic-based framework for mobile multi-agent systems. In *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 754–759, 10 2003. Boston, Massachusetts, USA.

[7] D. B. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[8] Napster. http://www.napster.com/.

[9] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. Edutella: A p2p networking infrastructure based on rdf. In *Proceedings*

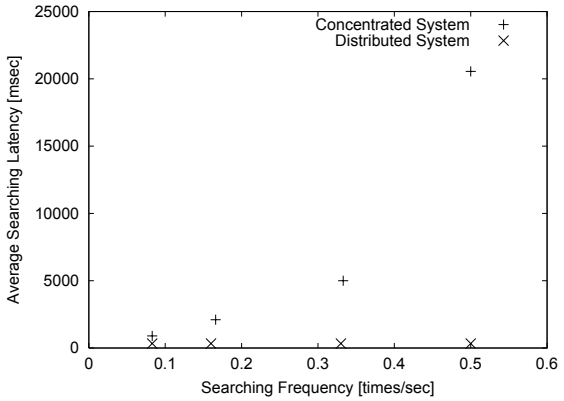

Figure 11. Comparison of concentrated and distributed systems in searching latency.

of the Eleventh International Conference on World Wide Web, pages 604–615. ACM Press, 2002.

[10] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. In U. Dayal, K. Ramamritham, and T. M. Vijayaraman, editors, *Proceedings of the 19th International Conference on Data Engineering*, pages 633–644. IEEE Computer Society, 2003.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[12] I. Satoh. Mobilespaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proceedings of IEEE International Conference on Distributed Computing Systems*, pages 161–168. IEEE Press, April 2000.

[13] P. Tarau. Inference and computation mobility with jinni. In K. Apt, V. Marek, and M. Truszczynski, editors, *The Logic Programming Paradigm: a 25 Year Perspective*, pages 33–48. Springer, 1999.

[14] D. Winer. Xml-rpc specification. http://xmlrcp.com/spec.

[15] D. Wong, N. Paciorek, T. Walsh, and J. Dicelie. Concordia: An infrastructure for collaborating mobile agents. In *Proceedings of the First International Workshop on Mobile Agents*, volume 1219, pages 86–97. Springer-Verlag, 1997.